10/517198
DT05 Rec'd PCT/PTO 08 DEC 2004

Objections to the X-documents listed in the International Search Report concerning Patent Application PCT/EP024927, dated 21th March 2003

In the text that follows, the objections are explained in detail. The text references and figures (page numbers, figure numbers) which are used to base the objections upon are cited for each X-document separately.

X-Document : 'Kiyohara, T. et.al., Register connection : a new approach for adding registers into instruction set architectures, Computer Architecture News, May 1993'

This X-document describes register mapping via register mapping instructions. Although the paper distinguishes between core registers and extended registers, **the mapping is still a mapping from a register to another register within the register file** (see figure 1 and section 2.1 on page 248).

In symbolic machine code as described by the present invention however, instruction operands and destinations may specify symbolic variables. By definition, a symbolic variable **does not specify a register within some register file of the microprocessor but one or more entries (or memory locations) in some dedicated memory other than the register file, and where each of said entries holds a memory address.** The memory address holds the value of said symbolic variable may be stored to and loaded from. Hence the instance link table as described in the present invention cannot be a register (re-)mapping table.

More specifically, as explained in section 4 'detailed description of the preferred embodiment' of the present invention, symbolic variables represent a totally new concept of making a link between memory addresses on one hand and instruction operands and destinations on the other hand. Symbolic variables allow the microprocessor to execute machine code in which no explicit load/store instructions appear. That is, by means of symbolic variables, the microprocessor is able to

generate load/store operations dynamically by himself, by using the memory address stored in a dedicated memory which is addressed (or specified) via the labels (or identifier) symbolic variables. The detailed procedure of generating load/store operations dynamically without relying on explicit load/store instructions is explained by the example of symbolic machine code listed on page 19 of the present invention.

Therefore, not only is the concept of symbolic variable a novel one, but it is also impossible for a person trained in prior-art to derive it from existing concepts, e.g. from register mapping as described in said X-document.

X-Document : 'Postiff, M., et. Al., The store-load address table and speculative register promotion, proc. of the ACM/IEEE micro-architecture symposium 2000'

This X-document describes a load-store address table which is used to map memory addresses to registers in order to do speculative register promotion and to save load/store operations. Special map instructions initialize this load-store table by explicitly associating memory addresses to registers (section 2, 1st , 2nd and 3rd paragraph on page 236). The question is whether these map instructions can be seen as being identical to those instructions specified in the example of symbolic machine code of page 19 of the present invention, associating memory addresses to symbolic variables.

There are two fundamental differences between these map instructions. First, as mentioned before, a fundamental difference lies in the fact that, by definition, **symbolic variables do not specify registers and are not aliases for registers.** Therefore a map instruction associating a memory address to a symbolic variable has nothing in common with mapping instructions used to initialize said store-load address table in said X-document.

Second, **said load-store table is indexed associatively by memory address** (see last sentence of 1st paragraph of section 2 on page 236). In contrast, the map instructions associating a memory address to a symbolic variable specify a table **which is indexed via the labels of the symbolic variables** (see 1st paragraph on page 20 of the present invention).

Finally, said load-store table does not allow to retrieve load/store instructions from the machine code. In contrast, symbolic variables make it possible for a microprocessor to execute machine code containing no load/store instructions (see example of symbolic machine code on page 19 of the present invention). The microprocessor is now able to generate load/store operations dynamically by himself, by using the memory address stored in a memory which is addressed (or specified) via the label (or identifier) of a symbolic variable. The detailed procedure of generating load/store operations dynamically without relying on explicit load/store instructions is explained by the example of symbolic machine code listed on page 19 of the present invention.

Hence, not only is the concept of symbolic variable a novel one, it is also impossible for a person trained in prior-art to derive it from existing concepts, e.g. from a load-store table.

X-Document : Patent Application EP 0767424

Claim 1 on page 8 of patent application EP 0767424 claims :

- an alias entry table where each alias entry stores a memory address, a length value and a base address in some alias buffer memory
- alias load/store instructions which allocate alias entries of said alias entry table to address locations in said alias buffer memory by initializing an alias entry in writing a base address into said alias entry table

More specifically, claims 6. to 12. on page 9 and figures 3. to 8. on pages 13 and 18 of patent application EP 0767424 describe how said aliases, referring to said alias entry

table, may be specified and used in the instruction format of said alias and non-alias load/store instructions.

Therefore, the question arises whether aliases as claimed in patent application EP 0767424 are identical to the concept and definition of a symbolic variable as described by the present invention.

There are several fundamental differences. First, in the present invention, a symbolic variable may be specified as operand **and destination** of **any** instruction of the machine code, not only as operands of said alias and non-alias load/store instructions as claimed in patent application EP 0767424.

Second, an alias entry of said alias entry table as defined in patent application EP 0767424 may have its value stored **either** at the memory address as specified in said alias entry table **or** in the alias buffer memory at the base address specified in said alias table, **but never** in the register file. That is, an alias can never specify a value stored in the register file. This is in sharp contrast to the definition of a symbolic variable in the present invention because the value of a symbolic variable can by definition be stored **anywhere** in the memory hierarchy, **also in the register file**. The small example of symbolic machine code on page 19 of the present invention shows how symbolic variables v0 to v6 are used as instruction operands and destinations. The data flow analysis on pages 19-23 of the present invention describes how the microprocessor determines whether the values of said symbolic variables are stored in The register file or in the upper memory hierarchy.

Third, figures 4. and 6. on pages 14 and 16 of patent application EP 0767424 show that, besides the purpose of allocation and de-allocation as explained in figures 3. and 8. on pages 13 and 18 of patent application EP 0767424, a said alias is obviously used in a said alias load/store instruction for the **sole** purpose of transferring a value between said alias buffer memory and a register in the register file. Said value is found in said alias buffer memory by using the alias specified in the instruction format of said instruction and the based address associated to said alias. Said register is also

specified in the instruction format of said instruction. However, this is a purpose which is **totally different** from the purpose of symbolic variables as described in the present invention. In the present invention, symbolic variables follow the main purpose of getting rid of load/store instructions in the machine code by letting the microprocessor decide itself when to generate load/stores instructions dynamically during machine code execution. This purpose is explained in the small example of machine code of page 19 (and following pages) of the present invention.

This stresses again that the concept of symbolic variable a novel one, and that it is impossible for a person trained in prior-art to derive it from existing concepts, e.g. from aliases as defined in patent application EP 0767424.